Abstract and History

Abstract and History

Abstract

For years, people have devised different techniques for encrypting data while others have attempted to break these encrypted codes. For our project we decided to put our wealth of DSP knowledge to use in the art of steganography. Steganography is a technique that allows one to hide binary data within an image while adding few noticeable changes. Technological advancements over the past decade or so have brought terms like “mp3,” “jpeg,” and “mpeg” into our everyday vocabulary. These lossy compression techniques lend themselves perfectly for hiding data. We have chosen this project because it gives a chance to study several various aspects of DSP. First, we devised our own compression technique which we loosely based off jpeg. There have been many steganographic techniques created so far, which compelled us to create two of our own strategies for hiding data in the images we compress. Our first method, zero hiding, adds the binary data into the DCT coefficients dropped in compression. Our other method, which we called bit-o-steg, uses a key to change the values of coefficients that remain after compression. Finally, we had to find ways to analyze the success of our data hiding strategies, so through our research we found both DSP and statistical methods to qualitatively measure our work.

A Brief History of Steganography

Steganography, or “hidden writing” can be traced back to 440 BC in ancient Greece. Often they would write a message on a wooden panel, cover it in wax, and then write a message on the wax. These wax tablets were already used as writing utensils, so the hiding of a message in a commonly used device draws very little suspicion. In addition to use by the Greeks, the practice of steganography was utilized by spies in World War II. There were even rumors that terrorists made use of steganography early in 2001 to plan the attacks of September 11

Compression Framework

Compression

Compression Framework

There are many picture file formats to save images to, however much of the research in steganography is done using the JPEG format. JPEG is a very common and uses a relatively straightforward compression algorithm. Although there are several JPEG compression scripts written for MATLAB, customizing them for our purposes and getting the output to work with the JPEG format would have shifted the focus of our project from steganography to implementing JPEG compression. Thus we decided to implement our own custom image framework that would be similar to JPEG but much more straightforward.

Compression - Dropping the DCT Coefficients

Compression Algorithm

Dropping DCT Coefficients

Our framework and JPEG are both based around the discrete cosine transform. Just like with sound, certain frequencies in an image are more noticeable than others, so taking them out of the image doesn't change the image much. We used the 2D discrete cosine transform (DCT) as seen in equation 1 to take an image and converts it into the frequencies that make up the image, in other words it takes us into the frequency domain.
Equation:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos(\frac{\pi(2m+1)p}{2M}) \cos(\frac{\pi(2n+1)q}{2N})$$

$where:$

$0 \leq p \leq M-1$

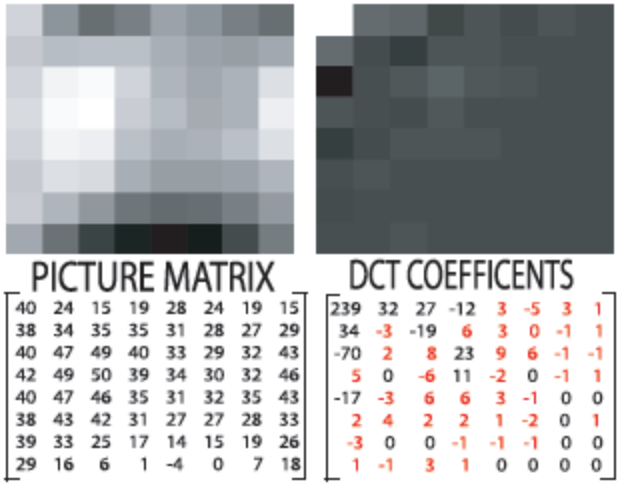
$0 \leq q \leq N-1$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, p = 0 \\ \sqrt{2/M}, 1 \leq p \leq M-1 \end{cases}$$

$$\alpha_q = \begin{cases} 1/\sqrt{N}, q = 0 \\ \sqrt{2/N}, 1 \leq q \leq N-1 \end{cases}$$

There are several transforms that could have been utilized to get the image into the frequency domain. The DCT, however, is a purely real transform. Thus, manipulating the frequencies is much more straightforward compared to other transforms. From here we could take the DCT of the entire image and then throw away frequencies that are less noticeable. Unfortunately this would make the image blurry and cause the image to lose edges. To solve this problem the image is divided into 8x8 blocks, to preserve the integrity of the image. To drop insignificant frequencies, JPEG compression utilizes

a quantization matrix. We simplified this process by using a threshold value and dropping frequencies below the threshold. Thus our compression algorithm models the basic functionality of the JPEG standard.



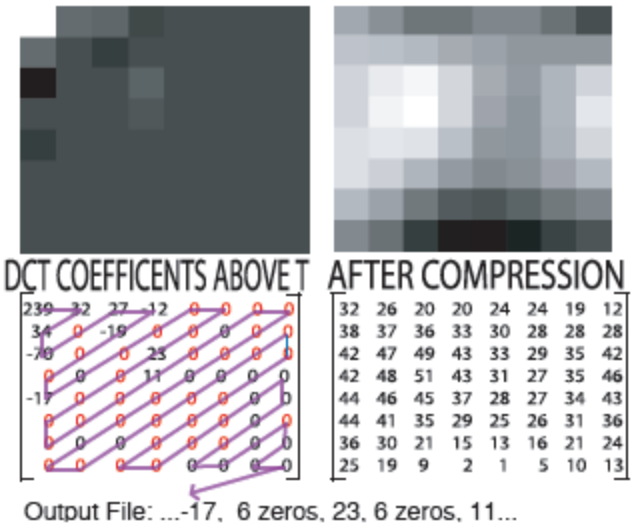
The result of taking the DCT. The numbers in red are the coefficients that fall below the specified threshold of 10.

Compression - Zeros Grouping

Compression Algorithm

Zeros Grouping

The second part to our image framework is zeros grouping. Just like the JPEG standard, the algorithm utilizes a zig-zag pattern that goes through each DCT matrix and creates a 64-length vector for each matrix. The advantage of the zig-zag pattern is that it groups the resulting vector from low frequencies to high frequencies. Groups of zeros are then replaced with an ASCII character representing how many zeros are represented within that group.



Zig-zag method traverses the matrix and vectorizes the matrix. After grouping zeros the resulting bitstream is sent to a file.

With this simple framework in place, we are able to model a real world image compression algorithm and focus on implementing steganography.

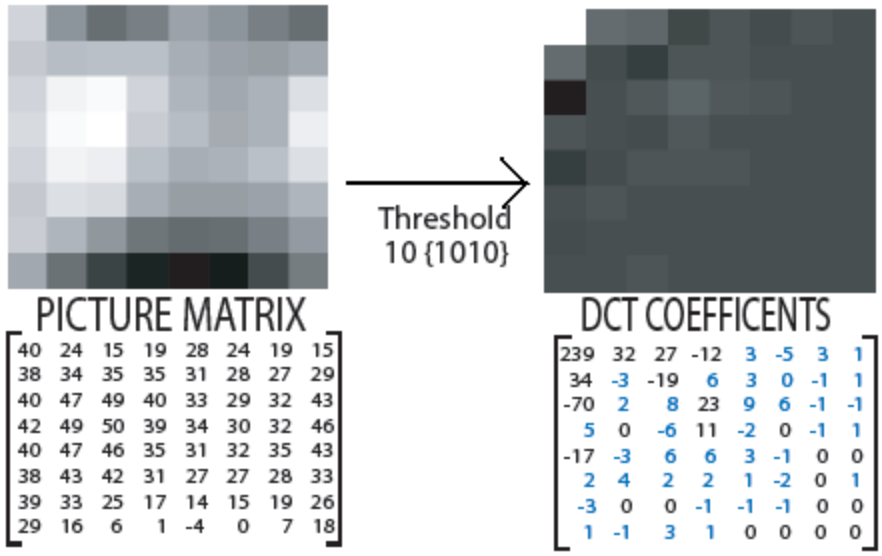
Zeros Hiding Method

Data Hiding Methods

Zero Hiding

Hiding Information

We arrived at our first data hiding method, which we called “zero hiding,” quite intuitively. If you recall, our compression algorithm removed the least important DCT coefficients. It follows, then, that we could put the bit stream we wish to hide back into these dropped coefficients without changing the image drastically. To do this though, there must be a way to distinguish a zero which resulted from a dropped coefficient and a coefficient that is zero. To do this, we ran the image through a modified compressor that, instead of dropping coefficients below the specified threshold, replaced them with either a plus or minus one, depending on the sign of the coefficient.



The DCT is taken and then each coefficient under

the specified threshold (10) will be dropped. These are coefficients are shown in blue in the picture on the right.

Next the hiding algorithm is given a binary data stream and the threshold value. The data stream is then divided up into words. However, the maximum decimal value of the word must be less than the threshold, since values over the threshold signify an important coefficient in the picture. We then increment each word's decimal value by one to avoid putting in zero valued coefficients, which would otherwise be indistinguishable from zero valued coefficients in the original image. We then go back to the original coefficients matrix and replace the ones with the new value of the data word, maintaining the sign throughout.

The dropped coefficients are replaced with words created from the data stream. The

IDCT is then taken, transforming the coefficient matrix back to a picture matrix.

Data Retrieval

To recover the hidden data the recovery script is given the threshold, and subtracts one from all DCT coefficients blow that threshold and tacks their binary values together, forming the original binary data.

Bit-O-Steg Method - Background

Data Hiding Methods

Bit-O-Steg

Previous Work and Background

In our research we found a steganographic method known as JSteg, created by Derek Upsham. The basic premise behind JSteg is that its algorithm hides the data sequentially within the least significant bits of the DCT coefficients (Niels and Honeyman). The problem with JSteg is that it is not very secure; there is no secret key with which it is encrypted. Therefore, anybody that knows an image contains data with JSteg hiding can easily retrieve the hidden message. Our second hiding method, which we have called bit-o-steg, improves upon the JSteg algorithm since we employ the use of a key when hiding the data.

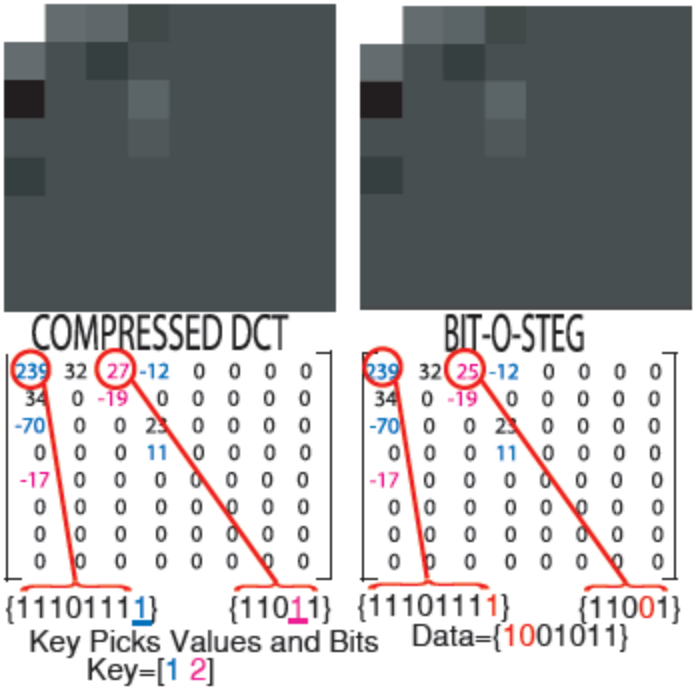
Bit-O-Steg Hiding

Data Hiding Methods

Bit-O-Steg

Hiding Information

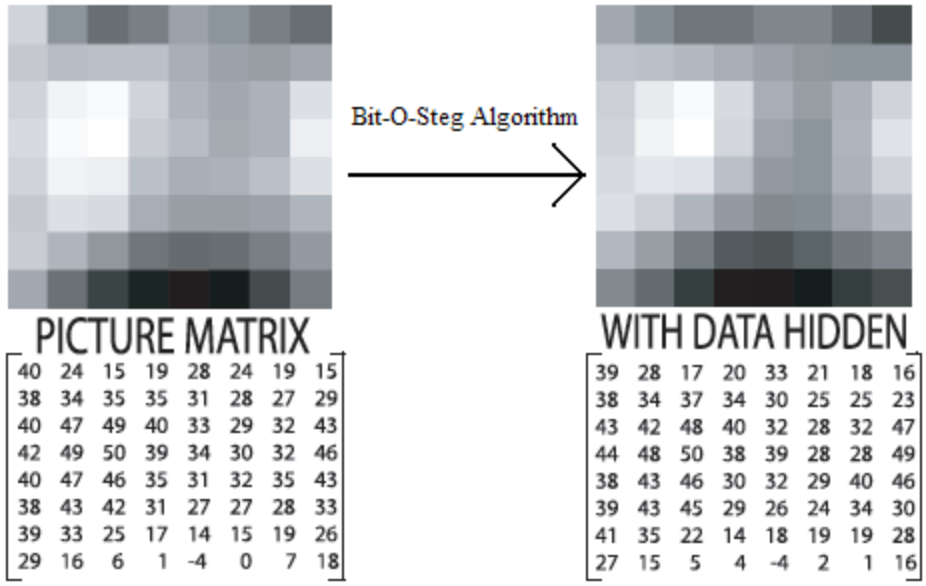
As you should recall, our zeros hiding method inserts data into the dropped coefficients of the DCT. The bit-o-steg algorithm hides data within the coefficients that were not dropped. The critical part of bit-o-steg is the key used to encrypt the data. This user defined key selects which nonzero coefficients to change and which bits to change within each coefficient. The simplest key would be a key of [1]. This would change each coefficient sequentially and change the last bit in the coefficient.



The key is what makes bit-o-steg

unique from other algorithms. Here a key of [1 2] is applied to hide the data.

As you can see in figure 1, we chose a key of [1 2]. The key will select the first coefficient and its least significant bit and input the first bit of the hidden data into that coefficient bit. Then the key will count two coefficients and take the second least significant bit and repeat the hiding process. Since this is the end of the key, it repeats, selecting the next coefficient. The length of this key has no real bound, but it must ensure that all data is hidden before reaching the last DCT coefficient in the image. There is, however, a range of values that must be selected for the key to work. Since the key alters bits, values between one and eight must be used. However, if larger values are used, it will alter the image greatly since it changes more and more significant bits.



Minimal changes have been made to the picture matrix after the application of the bit-o-steg algorithm

Retrieving the Data

Retrieving the data is impossible unless you have the special key used to hide the data. Once you get the key you simply reverse apply the key, extracting rather than inputting the bits and reconstruct your hidden data stream from those bits.

Importance of Steganalysis

Steganalysis

Importance of Steganalysis

Image steganalysis is the science of analyzing images in order to discover methods of discovering and detecting hidden messages and data within the images. Statistical digital signal processing is often used in order to detect data within images.

It is important to detect hidden messages within the images. On the steganography side, this is important in order to find methods in order to improve the algorithm implementing steganography. By exposing the flaws to the algorithm, the user can further improve the algorithm in order to make it more difficult to detect whether or not data is hidden in the images.

Steganalysis is also especially important in the security aspect, namely monitoring a user’s communication with the outside world. In the age of Internet, images are sent via email or by posting on websites. Detecting whether or not data is hidden in the images will allow the monitor to further analyze the suspicious images in order find what the hidden message is.

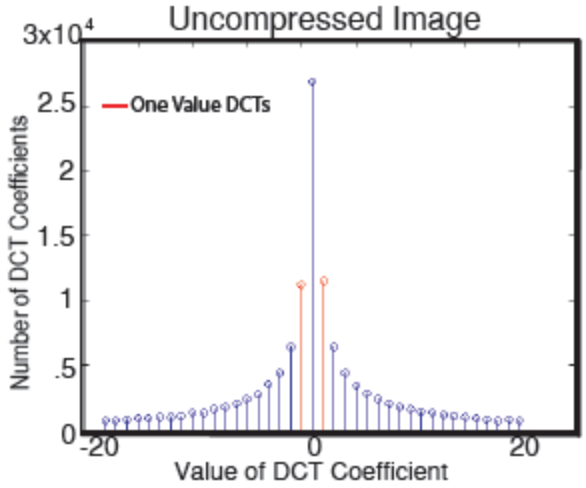
Can you tell if there is hidden data?

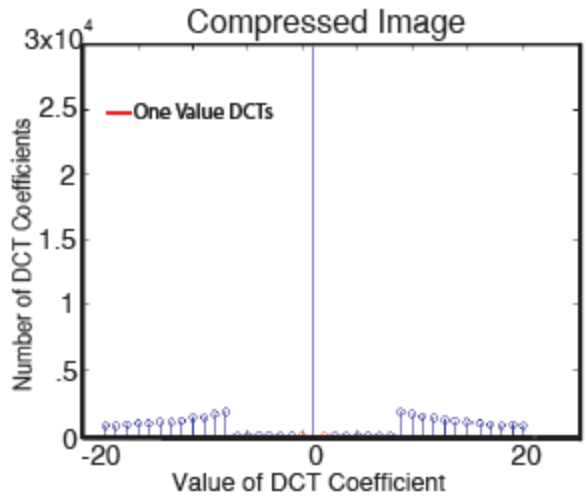
Steganalysis - Zeros Hiding Detection

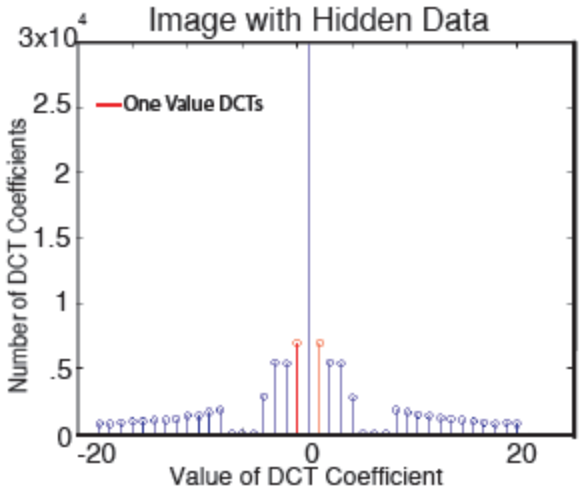
Steganalysis

Zeros Hiding Detection

In order to find data hiding with our zeros hidden method, we first analyzed the histogram of the DCT coefficients of an uncompressed image, compressed image without data, and compressed image with hidden data. The histogram of the DCT coefficients reveals the number of times each DCT coefficient value appears within the DCT matrix. From the analysis of an uncompressed image (Figure 1), the histogram has a smooth curve. In the histogram of compressed image (Figure 2), values before the threshold are dropped. Therefore, those values dropped to zero in the histogram. The histogram of compressed image with data (Figure 3) shows a similar shape to an uncompressed image. However, the values are much lower which makes sense since we are replacing the values that were originally going to be dropped with data. Therefore it is statically less likely to replace the dropped value with the same value.





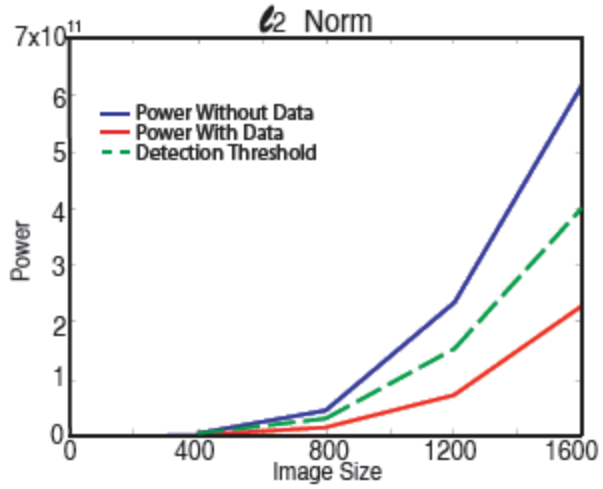


Therefore, after analyzing the histogram of the different types of images, we did an analysis of the l2 norm in the DCT matrix. If the analysis results in no power in the one valued DCT coefficients, it is a compressed image. This is due to the fact that ones are the minimum value that can be dropped. If there is power in the ones, then the image is either uncompressed or contains hidden data. The key difference between the two is the magnitude of the power in the ones. Statistically, it is less likely that every dropped coefficient gets replaced with a one. Therefore, the magnitude of the power in the ones in an image with data is lower than a compressed image. An image with hidden data will on average fall below a certain threshold. This threshold is dependent on the image size. Figure 4 shows the plots of the power without data, the power with data, and the threshold. Clearly, the

power without data is greater than the power with data. We found our detection program to have a 90% success rate but resulted in a false-positive 12% of the time.

Equation:

L2 Norm Equation



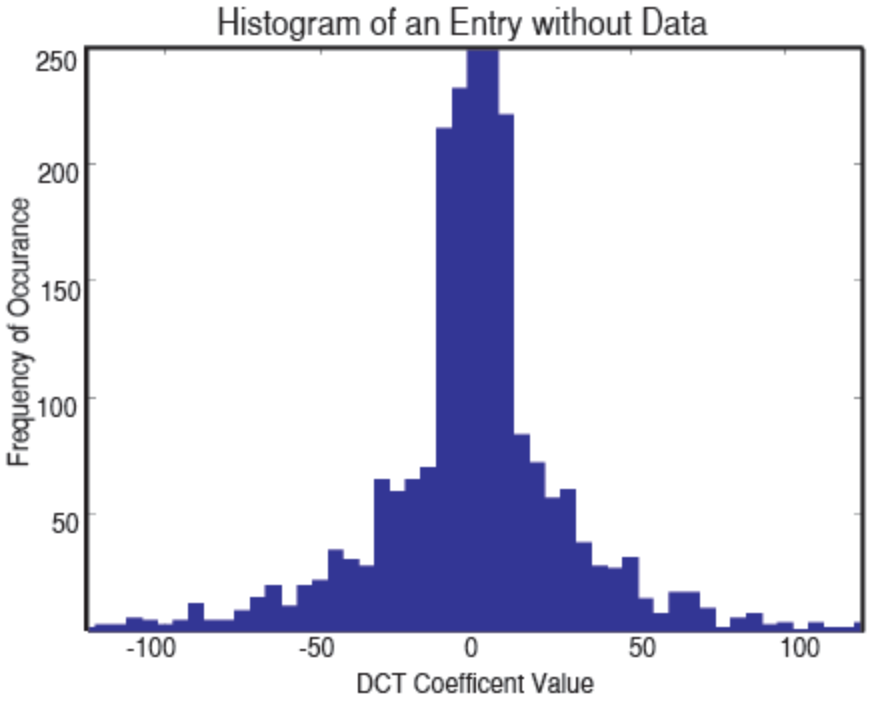
Steganalysis - Bit-O-Steg Detection

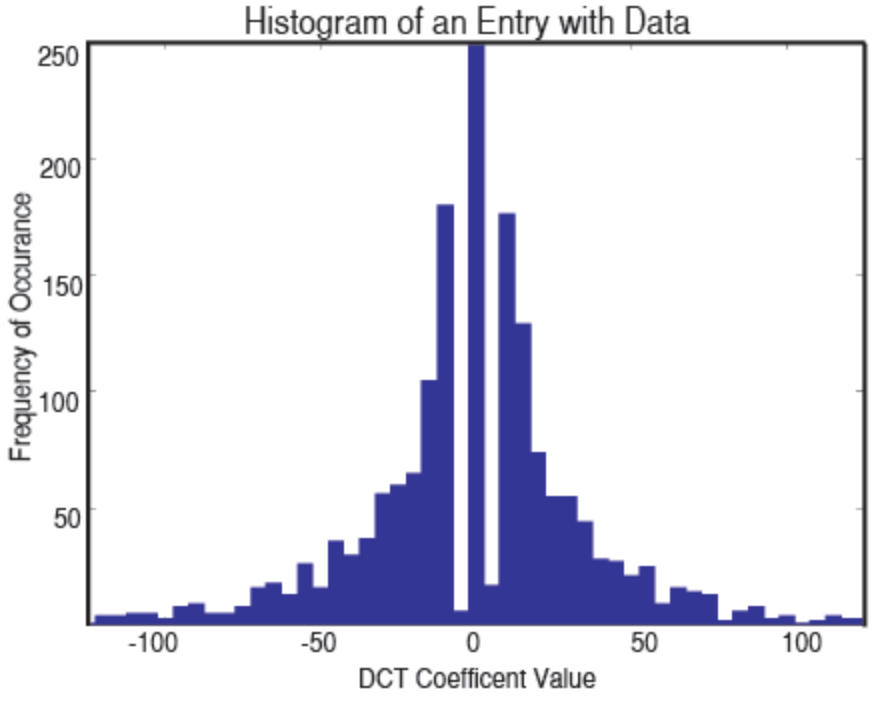
Steganalysis

Bit-o-steg detection

Due to the complexity of bit-o-steg, we turned to previous research to find a viable detection method. Each entry in the 8x8 blocks has a specific probability distribution. The distribution is found by looking at the values of that entry slot across the entire image. Figure 1 shows a histogram of an entry without data. The histogram looks at the DCT coefficient value and counts how often that value appears within that entry slot. Figure 2 shows a histogram of an entry with data. Comparing the two figures, there is a sudden drop around the 0 value in the histogram of an entry with data. The histogram of an entry with data also appears to smooth out.

These distributions are defined by their own characteristic functions. The bit-o-steg hiding distorts that distribution by randomly changes certain entries thus altering the function. Using the inner product, we could test for a match between the characteristic function and the suspect image's probability distribution. Unfortunately, the distribution functions vary based on the subject of the picture. Furthermore, we lack the statistical background necessary to classify these distributions and properly identify the characteristic functions. Thus, implementing bit-o-steg detection proved to be beyond the scope of this project.


Histogram of an Entry without Data
250
200
150
100
50
Frequency of Occurance
-100
-50
0
50
100
DCT Coefficent Value


Histogram of an Entry with Data
250
200
150
100
50
Frequency of Occurance
-100
-50
0
50
100
DCT Coefficent Value

Future Considerations and Conclusion

Future Work

Due to time and computing limitations, we could not explore all facets of steganography and detection techniques. As you saw, we studied the power in our pictures to test for hidden data. Another method which we were unable to explore was to analyze the noise of the pictures. Adding hidden data adds random noise, so it follows that a properly tuned noise detection algorithm could recognize whether or not a picture had steganographic data or not.

Conclusion

We explored several steganography techniques and the various detection algorithms associated with them. By using the properties of the DCT and our understanding of the frequency domain we developed the zeros hiding method. Zeros hiding proved to be easier to analyze than bit-o-steg and can hide significantly more data. Unfortunately its ease of detection makes it a less secure method. After researching various techniques already implemented, we chose to improve upon one, thus creating our bit-o-steg method. Bit-o-steg can only hide data in coefficients that were not dropped, thus limiting the amount of data we can hide. However, it greatly enhances the effectiveness of the steganography since it uses a key, making it much more challenging to detect. In the end we found both effective, but the complexity of bit-o-steg makes it more promising. Detection of our methods was critical to the breadth of our project. By investigating the power in various components of our images we discovered how to detect data hidden via the zero hiding method. Detecting bit-o-steg required us to draw on past steganography research and statistically analyze the effects of this type of data hiding. The methods and accompanying detection schemes we developed broadened our understanding of steganography, which, unlike

encryption, allows secret data to be traded hands without raising an eyebrow.

Works Cited

Works Cited

Cabeen, Ken, and Peter Gent. “Image Compression and the Discrete Cosine Transform.” College of the Redwoods. <http://online.redwoods.cc.ca.us/instruct/darnold/laproj/Fall98/PKen/dct.pdf >

Johnson, Neil F., and Sushil Jajodia. “Exploring Steganography: Seeing the Unseen.” George Mason University. <http://www.jjtc.com/pub/r2026.pdf>

Johnson, Neil F., and Sushil Jajodia. “Steganalysis: The Investigation of Hidden Information.” George Mason University. <http://ieeexplore.ieee.org/iel4/5774/15421/00713394.pdf?tp=&arnumber=713394&isnumber=15421>

Judge, James C. “Steganography: Past, Present, Future.” <http://www.sans.org/rr/whitepapers/stenganography/552.php>

Provos, Niels, and Peter Honeyman. “CITI Technical Report 01-11. Detecting Steganographic Content on the Internet.” University of Michigan. <http://www.citi.umich.edu/techreports/reports/citi-tr-01-11.pdf>

Provos, Niels, and Peter Honeyman. “Hide and Seek: An Introduction to Steganography.” University of Michigan. <http://niels.xtdnet.nl/papers/practical.pdf>

Sallee, Phil. “Model-based Steganography.” University of California, Davis. <http://redwood.ucdavis.edu/phil/papers/iwdw03.pdf>

Silman, Joshua. “Steganography and Steganalysis: An Overview.” <http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=241>

Wang, Huaiqing, and Shuozhong Wang. “Cyber warfare: Steganography vs. steganalysis.” Communications of the ACM, Volume 47, Number 10.

<http://acmqueue.com/modules.php?name=Content&pa=showpage&pid=241>

Steganography Matlab Code

detect_data_power.m Detects if the given image has data hidden in it with the zeros hiding method

hidden_zeros_read.m Reads data hidden by zeros hiding method from an image

image_hist.m Creates histogram of DCT coefficients

image_stat.m Determines the threshold value for zero hiding detection

invimageproc.m Takes tiled matrix and converts to image matrix

jvector.m Traverses 8x8 block using the zig-zag pattern

mat2DCEB.m Compresses image and returns tiled matrix

readdata.m Reads in binary data from a file

secread.m Reads in data hidden by bit-o-steg from an image

signed_mat2DCEB.m Modified compressor used in zeros hiding

std_stegcompress.m Groups zeros together and saves image

stegcompress.m Takes a compressed image and hides data in it using bit-o-steg

writedata.m Writes binary data to a file

Group Members

Group Bio

Elliot Ng, Jones 2007

Bryan Grandy, Brown 2007

Charlie Ice, Brown 2007

Danny Blanco, Jones 2006